# REKURSION, SUCHEN, DATENSTRUKTUREN

## Rekursion („Selbstaufruf“)

**Lösungsmethoden**

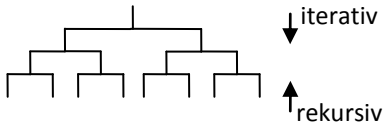
| | |
|---|---|
| **iterativ** | schrittweise nacheinander |
| **rekursiv** | ineinander geschachtelt |

**Rekursiv definierte Folgen**

| | |
|---|---|
| **Rekursionsbasis** (Anfangsbedingung) | $a_0 = \#$ |
| **Rekursionsvorschrift** (Folgeglieder) | $a_n = f(a_{n-1})$ |

### Prinzip der Rekursion

Unterteile in Teilprobleme
Löse Baum von unten




**Nachteile** erhöhter Speicherbedarf
**Permutationen** {A, B, C} = ABC, ACB, BAC, BCA, …

**Grundstruktur (im Pseudocode)**

```
Programm Löse(Problem)
BEGIN
   IF(Problem einfach) THEN
      Löse //Rekursionsbasis
   ELSE
      Unterteile //Rekursionsvorschrift
      Löse(Teilproblem)
   END
END
```

### Backtracking (try and error / Versuch und Irrtum)

| Idee | Eigenschaften | Beispiele |
|---|---|---|
| • vor und zurückgehen bis Lösung gefunden<br>• nach Versuch und Irrtum | • oft Rekursiv | • Labyrinth<br>• Springerproblem<br>• Damenproblem |

## Suchen

| | |
|---|---|
| **Internes Suchen** | Alle Daten passen in den Speicher |
| **Externes Suchen** | Auslagerung auf Datenbanken |

**In Arrays und verketteten Listen**

| | | Array (=Reihung) | | | Listen | |
|---|---|---|---|---|---|---|
| | | unsortiert | sortiert | | unsortiert | sortiert |
| | | linear | linear | binär (teile und herrsche) | linear | linear |
| **Suchen** | Zeitkomplexität | $O(n)$ | $O(n)$ | $O(\log n)$ | $O(n)$ | $O(n)$ |
| | Speicherkompl. | $O(1)$ | $O(1)$ | $O(n)$ | $O(n)$ | $O(n)$ |
| **Einfügen** | Zeitkomplexität | $O(1)$ | $O(n)$ | $O(\log n)$ | $O(1)$ | $O(n)$ |

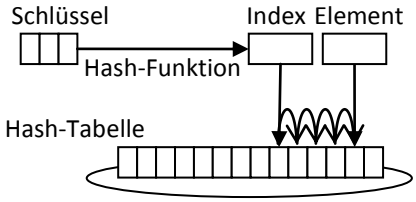
**Hashtable**

**Ausgangslage:** Objekte werden in einer Tabelle anhand von Schlüsseln (key, K) abgelegt.

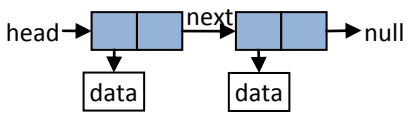
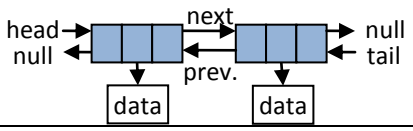
**Problem:** Welche Datenstruktur??
Array -> Grösse unbekannt
Verkettete Liste -> langsam

**Lösung:** Hashtable
1. Einfügen: Aus Schlüssel wir ein Index berechnet
2. Solange an nächste Stelle gehen bis jene frei ist.

**Idee**: Mischung von Array + verketteter Liste




## Datenstrukturen („Behälterklassen“)

| Reihung (=Array) | Einfach verkettete Listen | Doppelt verkettete Listen |
|---|---|---|
| Wie ein Papierstapel<br>Zugriff nur auf oberstes Element | Variablen sind nur Referenzen auf Objekte (Ausnahme: Basisdatentypen)<br>dynamisch (wachsen, schrumpfen); sind rekursiv | |
|  <br>**Nachteil** Grösse festlegen,<br>**Vorteil** Schneller Zugriff |  |  |
| **Einfügen:** `put()`<br>**Löschen:** `get()` | **Einfügen:** `insert()` `//vorne`<br>`append()` `//hinten`<br>**Löschen:** `delete()` | |